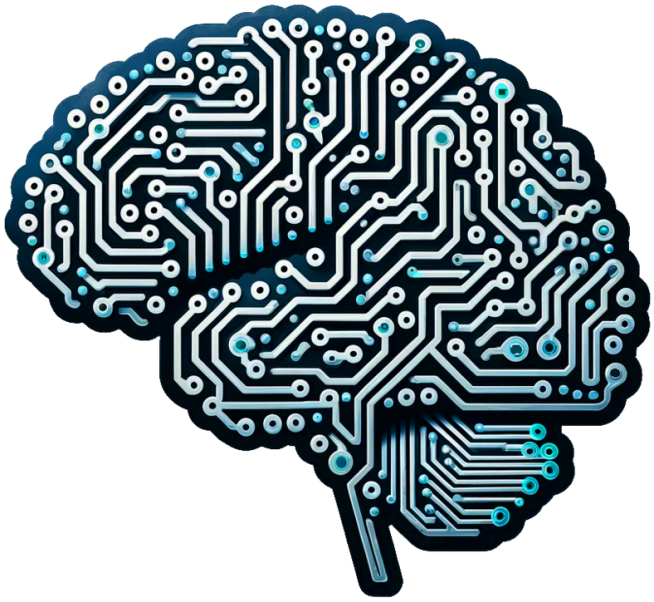
Introducción a ML y Generative AI



Taller de programación #1

Descripción

Este documento contiene información relacionada con las actividades de prácticas del primer taller del curso de introducción a ML y Generative AI. En este taller, empezaremos con una pequeña revisión de las herramientas que usaremos en el curso. A continuación, mostramos cómo ejecutar instrucciones básicas de Python, instalar librerías externas con pip, y proporcionamos los pasos iniciales para trabajar con el dataset Titanic. Por favor, tenga en cuenta que el taller se ha realizado en una computadora con el sistema operativo Windows 10.

1. Revision

1.1 Command Prompt (Simbolo del sistema)

Permite al usuario interactuar directamente con el sistema operativo, los comandos son en forma de línea de textos, antiguamente se usaba para controlar el sistema puesto que no existía mouse, control de manejo de ventanas o GUI.

1.2 Editor de código fuente

Es un programa que permite crear o modificar archivos de forma digital. Pueden decidir usar el editor de código fuente de su preferencia para este y los próximos talleres. En caso de no haber usado un editor de código anteriormente les recomendamos Visual Studio Code. Si desean descargarlo pueden hacerlo usando el siguiente enlace: https://code.visualstudio.com/.

Nota: Visual Studio Code contiene una terminal integrada, Puedes usar esta terminal si lo deseas una vez hayas instalado Python.

1.3 Entorno virtual

Los entornos virtuales son herramientas que permiten a los desarrolladores crear un espacio aislado en el que se pueden instalar las dependencias (a menudo bibliotecas y paquetes) requeridas para un proyecto específico sin interferir con otras instalaciones de software.

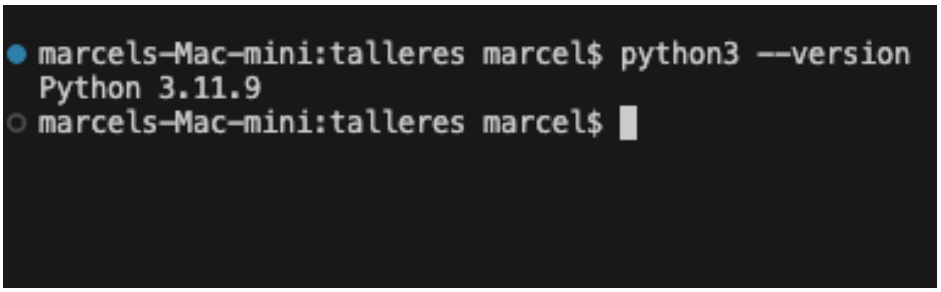
1.4 Paquetes y módulos

Un paquete en Python es una forma de organizar módulos relacionados. Un módulo es un archivo que contiene código Python, y un paquete es un directorio que puede contener varios módulos y otros paquetes. Dentro del campo del aprendizaje automático e inteligencia artificial, los científicos y desarrolladores usan paquetes como Scikit-learn, TensorFlow, Pandas, entre otras.

2. Python

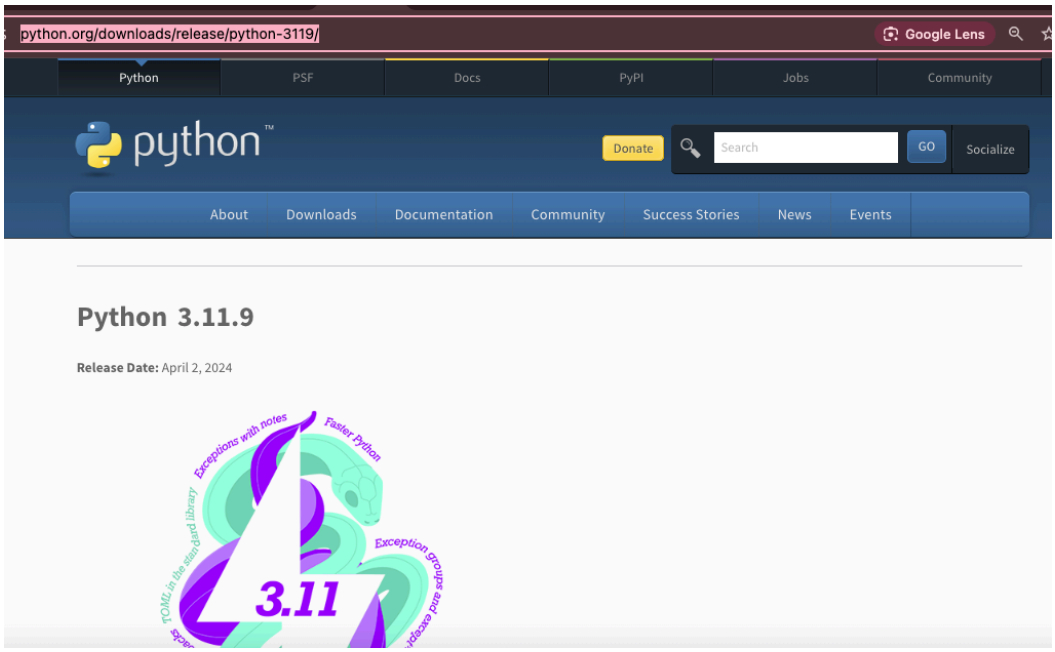
2.1 Confirmar que Python está instalado

Recomendamos que confirmes si tienes la versión de Python 3.11, abriendo una terminal o símbolo del sistema e ingresando el comando python --version, o si hay múltiples versiones de Python instaladas, python3 --version. Si tienes la versión de Python 3.11, puedes ir directamente a la sección 3.



2.2 Instalar Python

Para obtener la versión de Python 3.11 usando el siguiente enlace
https://www.python.org/downloads/release/python-3119/



Desciende hasta el final de la página web, dirígete hacia la sección archivos y selecciona el instalador para tu sistema operativo (si estás confundido con el instalador, por favor levanta tu mano durante la hora de taller)

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG	Sigstore
Gzipped source tarball	Source release		bfd4d3bfeac4216ce35d7a503bf02d5c	25.3 MB	SIG	.sigstore
XZ compressed source tarball	Source release		22ea467e7d915477152e99d5da856ddc	19.2 MB	SIG	.sigstore
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	fa29f456feb6b5c4f52456a8b8ba347b	42.8 MB	SIG	.sigstore
Windows installer (64-bit)	Windows	Recommended	e8dcd502e34932eebcaf1be056d5cbcd	25.0 MB	SIG	.sigstore
Windows installer (32-bit)	Windows		2a1d1ac2d8a0aa847515f9dd121ccbb7	23.8 MB	SIG	.sigstore
Windows installer (ARM64)	Windows	Experimental	328d93f71cb078965e4cfa2eb2663fa1	24.3 MB	SIG	.sigstore
Windows embeddable package (64-bit)	Windows		6d9aa08531d48fcc261ba667e2df17c4	10.7 MB	SIG	.sigstore
Windows embeddable package (32-bit)	Windows		31e7648158376e92a4463aa6f22a78e1	9.6 MB	SIG	.sigstore
Windows embeddable package (ARM64)	Windows		8611b6aa35483ab1c61d45e0d9f2de0d	10.0 MB	SIG	.sigstore

Dependiendo del sistema operativo, deberás descargar diferentes archivos. En el caso que uses un sistema diferente a Windows. Por Favor, envíanos un mensaje para ayudarte con el proceso de instalación.

Para Windows: Una vez que el ejecutable ha sido descargado, es posible iniciar el proceso de instalación haciendo doble clic izquierdo en el ejecutable. Durante la instalación, asegúrate de marcar la casilla "Add python.exe to PATH" y seleccionar "Install Now" para continuar con la instalación.

Una vez el instalador finalice, comprueba que Python está instalado ejecutando los comandos mencionados en la section 2.1

Nota: Es posible que necesites abrir una nueva terminal para comprobar que Python ha sido instalado, o que reinicies tu computador.

2.3 Organización de código

Con el fin de organizar nuestro código, crearemos una carpeta en el escritorio. Para esto ejecute en su símbolo de sistema > cd c:\Users\(nombre_de_usuario)\Desktop > mkdir curso_ml > cd curso_ml

2.3 Configurar un entorno virtual (Opcional)

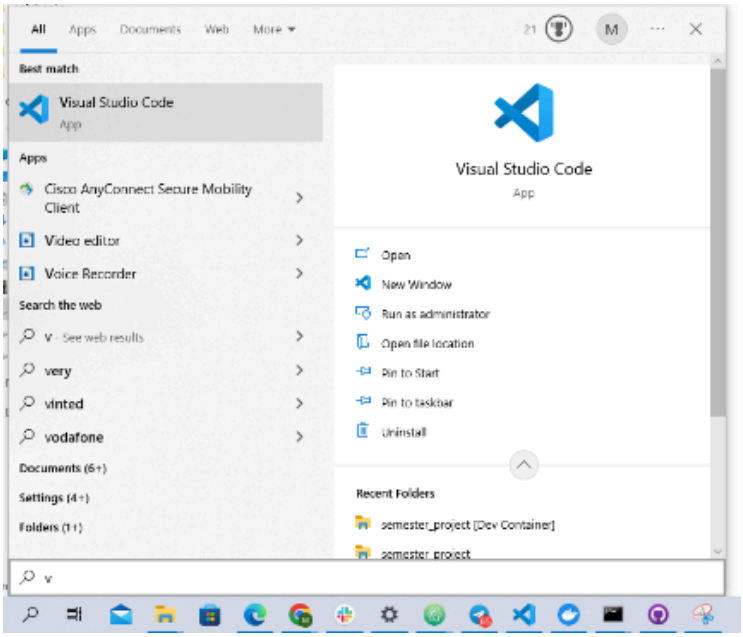
Crear un entorno virtual es considerado una buena práctica en el desarrollo de software, ya que evita conflictos de dependencias y problemas de compatibilidad. Python nos permite crear estos entornos usando el siguiente comando: **python -m venv intro_ml** dentro de “curso_ml". Al hacer esto podrás observar una nueva carpeta en el directorio con el nombre de intro_ml. Para iniciar el entorno desde la terminal en la carpeta curso_ml usa el siguiente comando:

Windows: intro_ml\Scripts\activate.bat
Unix or MacO: source intro_ml/bin/activate

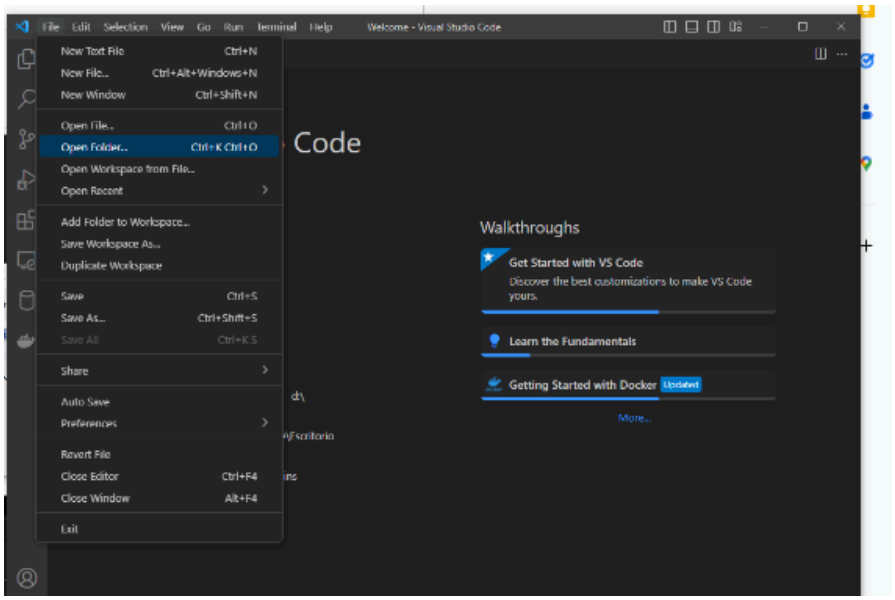
```
marcels-Mac-mini:curso_ml marcel$ source intro_ml/bin/activate
(intro_ml) marcels-Mac-mini:curso_ml marcel$
(intro_ml) marcels-Mac-mini:curso_ml marcel$
(intro_ml) marcels-Mac-mini:curso_ml marcel$
```

3. Mi primer programa

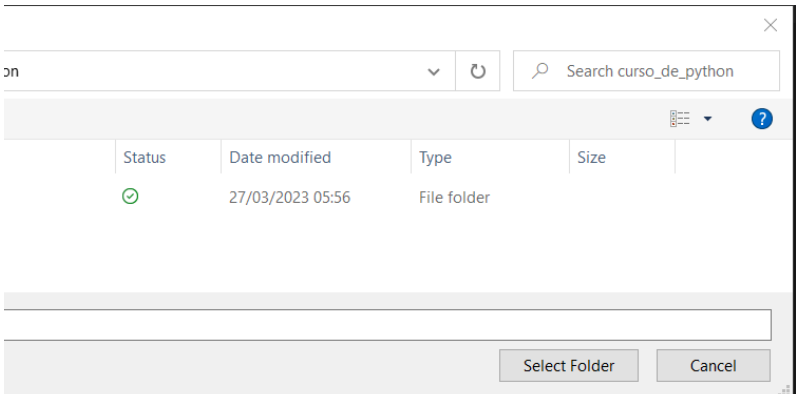
Empezaremos nuestro tutorial abriendo el editor de código fuente “Visual Studio Code” que nos permitirá crear nuestro primer programa. Una vez lo hayas descargado e instalado lo puedes abrir como cualquier otra aplicación de Windows.

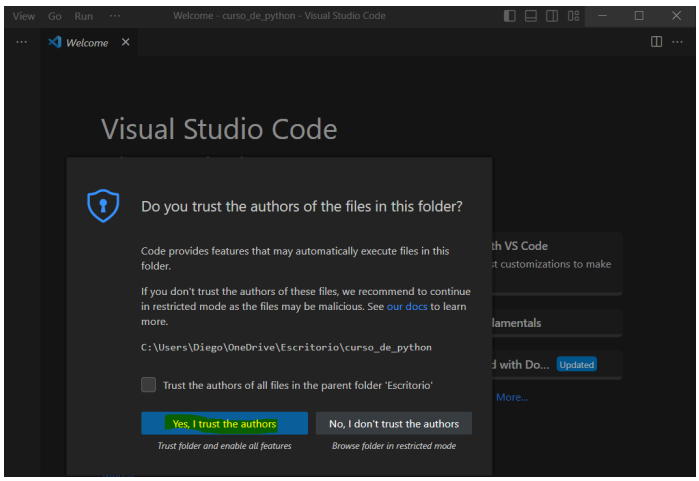


Una vez en el editor, podemos abrir la carpeta ‘curso_de_ml’ con el material del curso haciendo click en File y luego Open Folder como se muestra en la siguiente imagen:

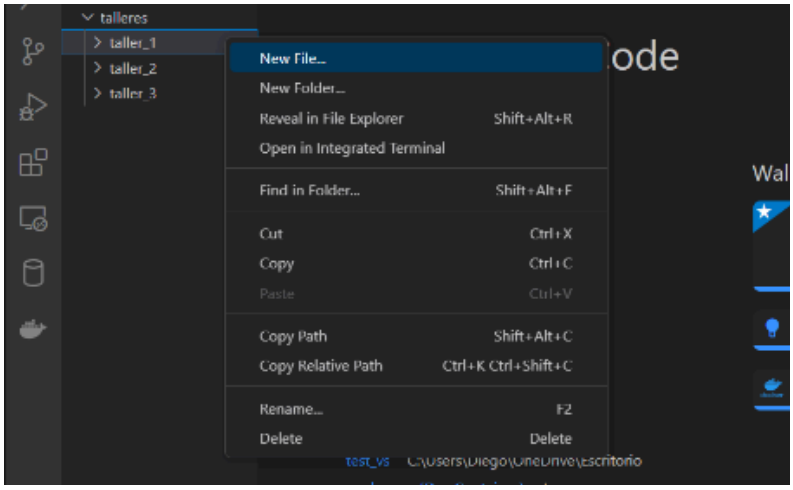


Una vez nos hayamos dirigido a la ruta de nuestra carpeta debemos hacer clic en ‘seleccionar carpeta’ y aceptamos confiar en los autores de la carpeta. Esto nos permitirá visualizar y explorar sus contenidos como se muestra en las siguientes imágenes:

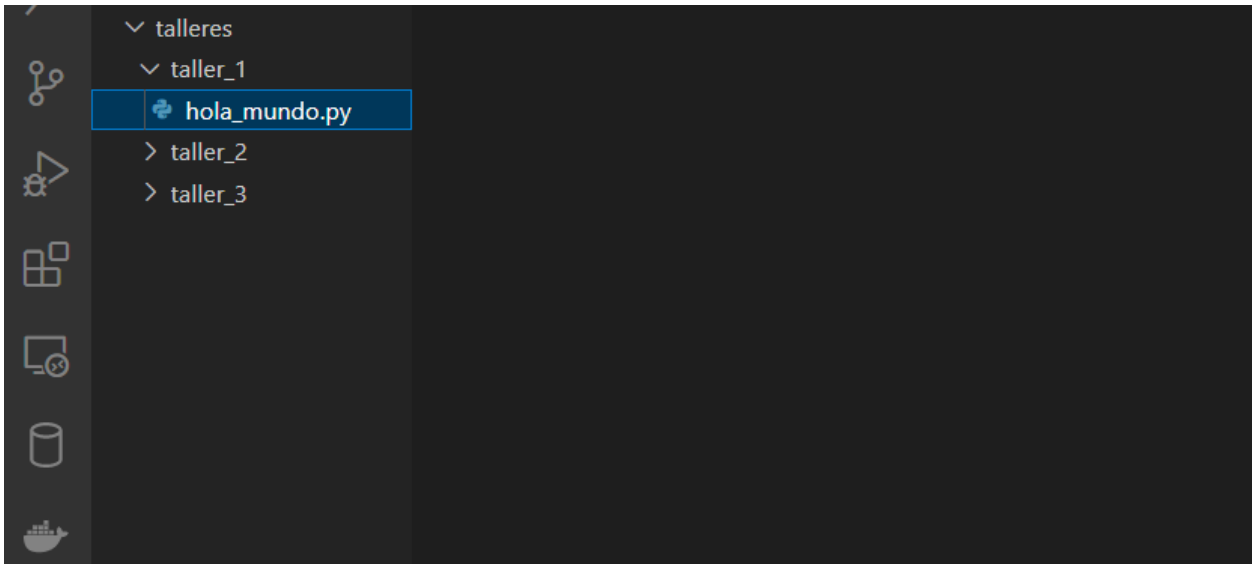




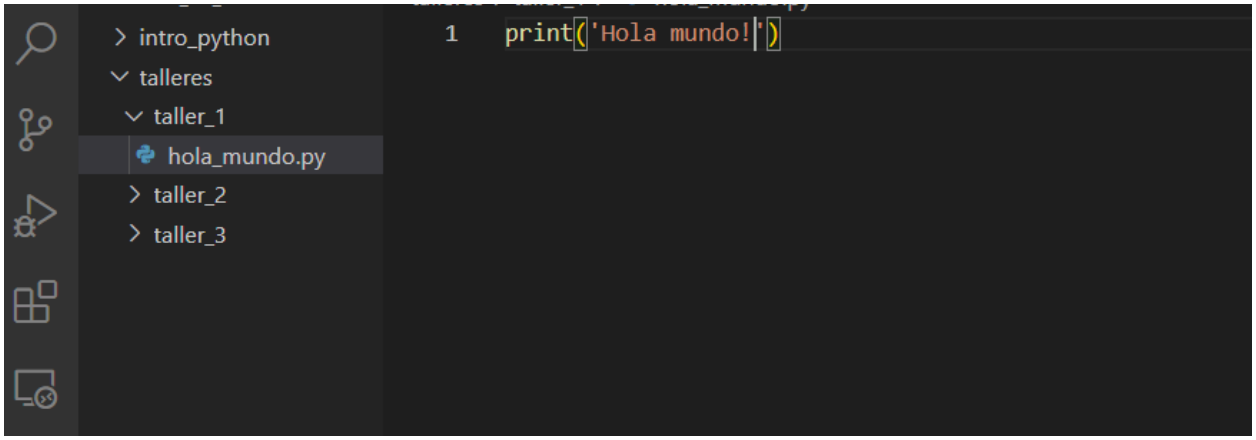
Se puede añadir un nuevo archivo en una carpeta en específico al hacer clic derecho sobre esta y seguido seleccionamos ‘New File’. En la siguiente imagen se muestra cómo creamos una nuevo archivo en la carpeta ‘taller_1’:



A continuación Visual Studio Code nos preguntará cómo nombrar al archivo, en nuestro caso lo llamaremos ‘hola_mundo.py’. Recuerda seleccionar la terminación ‘.py’ ya que esto permite que el editor identifique nuestro archivo como un archivo de Python.

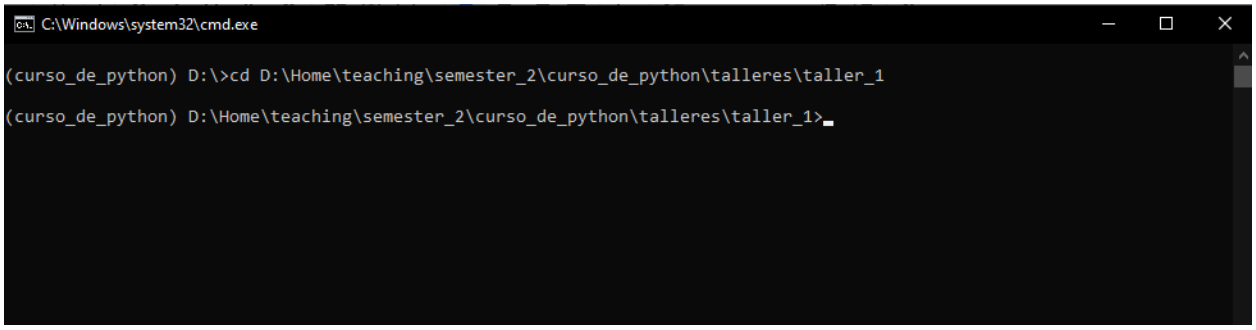


Esto hará que Python cree y abra el nuevo archivo donde podemos escribir sintaxis de Python. Para nuestro primer programa haremos que la consola muestra un mensaje de bienvenida, para esto usaremos una palabra especial en python llamada print (print nos permite enviar parámetros e imprimirlos linea por linea)

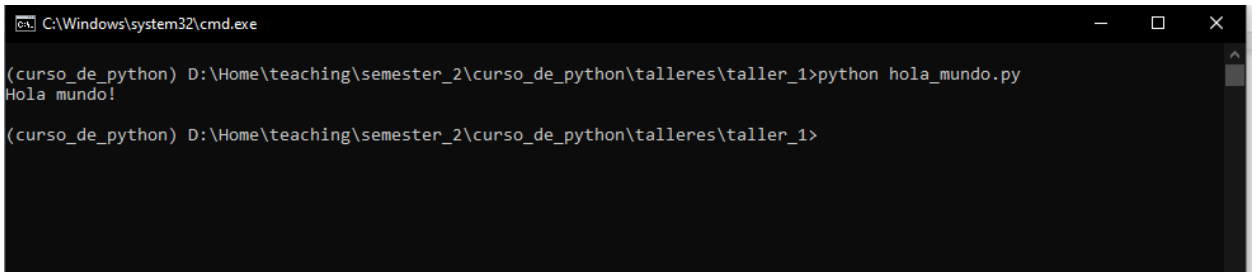


Para guardar los cambios hechos al documento puedes simplemente ocupar el atajo ‘CTRL + S’ o dirigirte a la pestaña ‘File’ y luego selecciona ‘Save’.

Para poder correr el archivo que hemos escrito debemos primero empezar nuestro entorno de desarrollo siguiendo los pasos explicados en la parte 2.1.2 de este taller. Una vez que se abra la terminal debemos dirigirnos a la ruta donde están nuestros archivos. Para eso podemos ocupar el comando ‘**cd**’ que nos permite movernos a través de directorios.



Desde el símbolo de sistema, llamaremos al interpretador de Python usando ‘python hola_mundo.py’’



4. Instalando librerías

Instalar librerías en Python es trivial gracias a herramientas como pip. Vamos a instalar un paquetes populares en el sector de análisis de datos (pandas, numpy, scikit-learn, y matplotlib) con ayuda del comando **pip install <nombre del paquete>** dentro de tu terminal o **pip3 install <nombre del paquete>** si tienes múltiples versiones .

Comando:

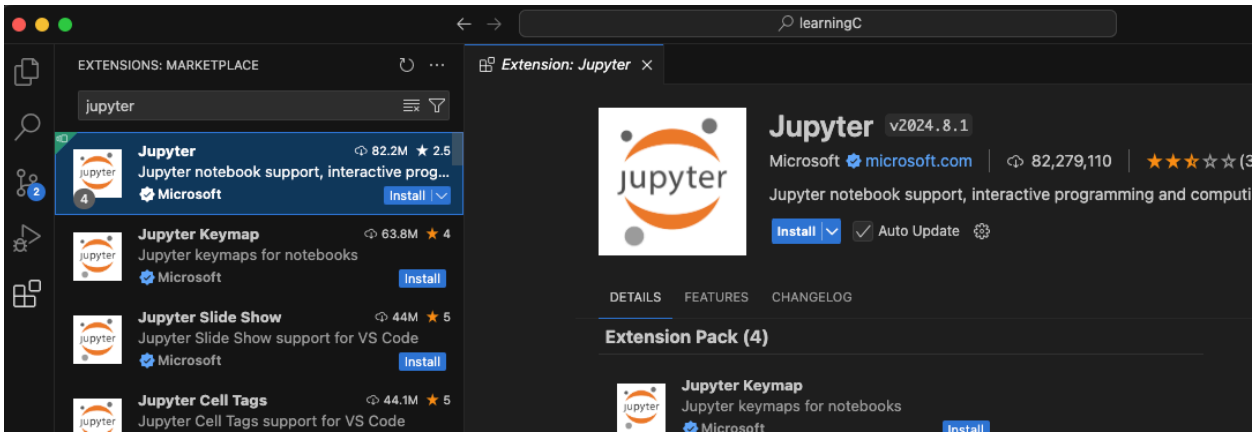
```
pip install pandas numpy scikit-learn matplotlib
```

```
marcels-Mac-mini:talleres marcel$ pip3 install pandas numpy scikit-l
earn matplotlib
Collecting pandas
  Downloading pandas-2.2.2-cp311-cp311-macosx_10_9_x86_64.whl.metadata (19 kB)
Requirement already satisfied: numpy in /Library/Frameworks/Python.framework/V
/site-packages (2.1.1)
Requirement already satisfied: scikit-learn in /Library/Frameworks/Python.fram
hon3.11/site-packages (1.5.2)
Requirement already satisfied: matplotlib in /Library/Frameworks/Python.framew
```

Nota: Para evitar instalar los paquetes uno por uno, podemos añadir todos en una sola línea.

5. Familiarízate con un editor de código:

Usa un editor de código como Visual Studio Code o Jupyter Notebooks, que permiten editar y ejecutar código Python. Si usas Visual Studio Code, también es recomendable **instalar la extensión de Jupyter.**



6. Cargar un dataset usando un archivo CSV:

Importar las librerías necesarias: Antes de cargar un archivo CSV, importa las librerías que manejarán los datos:

Comando:
import pandas as pd

1. **Descarga el Titanic dataset de Kaggle:**
   - https://www.kaggle.com/datasets/yasserh/titanic-dataset. (Puede que necesites crear una cuenta primero).

2. **Localizar el archivo CSV:** Asegúrate de conocer la ruta exacta del archivo CSV en tu sistema o proyecto. Si el archivo está en el mismo directorio que tu código, puedes simplemente usar su nombre. Si está en otro lugar, deberás proporcionar la ruta completa.

3. **Cargar el dataset en un DataFrame de pandas:** Usar pandas es una de las formas más comunes de cargar datos desde archivos CSV. El siguiente código carga el archivo CSV en un DataFrame:
   Comando:
   df = pd.read_csv('ruta/del/archivo.csv')

4. **Verificar la carga de datos:** Es recomendable verificar si el dataset se ha cargado correctamente. Puedes revisar las primeras filas con:

   print(df.head())

5. **Revisar el tamaño del dataset:** Es útil saber cuántas filas y columnas tiene el dataset:

   print(df.shape) # (filas, columnas)

6. **Explorar el dataset:** Revisa la estructura de los datos para familiarizarte con las columnas y tipos de datos. Esto es crucial antes de proceder con el análisis o modelado de Machine Learning.

   print(df.info())

   print(df.describe())

7. **Buscar valores nulos:** Es importante saber cuántos valores faltantes hay en cada columna, ya que estos pueden afectar el análisis y modelado

   ```
   print(df.isnull().sum())
   ```

8. **Explorar las columnas categóricas:** Las columnas categóricas (como Sex, Embarked, Pclass) son importantes en el Titanic dataset. Puedes usar value_counts() para explorar la distribución de los valores en estas columnas:

   ```
   print(df['Sex'].value_counts())

   print(df['Embarked'].value_counts())

   print(df['Pclass'].value_counts())
   ```

9. **Visualizar la distribución de las variables numéricas**

   Es buena práctica visualizar la distribución de las variables numéricas como la edad, la tarifa del pasaje (Fare), etc. Puedes usar gráficos de histograma:

   ```
   df['Age'].hist(bins=30)

   plt.title('Distribución de Edad')

   plt.xlabel('Edad')

   plt.ylabel('Frecuencia')

   plt.show()
   ```

Marcel Mauricio Moran Calderon
marcel_moran41@hotmail.com

Ariel Ramos Vela
ariel.ramos97@gmail.com